

ErIIDE

An eclipse feature for erlang development

Eclipse

- Evolved from Visual Age
- 100% Open Source
- 100% java
- IBM
- Major platforms
(Windows, Mac, Linux)

ErIIDE

- A bunch of plugins for Eclipse
- Originally 100% java
- Code quantity: 60% java, 40% erlang

ErlIDE and OTP

- R13B or later
- Compiler
- JInterface
- special versions of `syntax_tools`, `edoc`,
debugger, scanner

ErlIDE Features 1

- Erlang editor
 - Syntax highlight
 - Indentation
 - Pop-up documentation
 - Outlining
- Integrated compiler
 - Error list
- Todo-lists

ErlIDE Features 2

- Navigation
 - Open declarations with ctrl-click
 - Outline with filters
 - Quick select of declaration
 - Erlang navigator
- Documentation hover
 - OTP documentation
 - Macro and record declarations
 - Comments above functions

ErlIDE Features 3

- Project support
 - New erlang projects
 - Combining erlang and other (Java, C)
 - Importing erlang projects
- Integrated debugger
 - Breakpoints
 - Single-stepping, step into, step out
 - Multiple erlang processes
 - Local variable inspection
 - Change values

ErlIDE Features 4

- Semantic highlighting
- Erlang searching
- Support for erlang tracing (TTB and DBG)

Coming attractions

- Improved documentation support (edoc)
- Incremental background compiling
- Full code formatting
- Refactoring with syntax error correction
- Other nice and useful features

Installing

- Download and install Eclipse, 3.5 or later
- Download and install OTP, R13B02 or later
- Fetch ErlIDE from within Eclipse
- Set up Erlang runtime preferences
- Done

Editing

- Bracket matching
- Auto-indentation
- Completion
- Editing preferences

Editor and outline

The screenshot displays the Eclipse IDE interface for Erlang. The main editor window shows the source code for `yaws_api.erl`. The code includes several functions: `eat_until`, `url_decode`, and `path_norm`. The `url_decode` function is currently selected and highlighted. The `Outline` view on the right lists the functions defined in the file, with `url_decode/1` expanded to show its signature `url_decode([Hi, Lo | Tail])`. The `Erlang console` at the bottom shows the current Erlang backend node as `erlang: 4d9de8e_jakob_erlide` and the shell prompt `(4d9de8e_jakob_erlide@gno-mac-5.du.uab.ericsson.se)1>`. The `Erlang Navigat` view on the left shows the project structure, including the `src` directory where the current file is located.

```
eat_until(L, U) ->
    eat_until(L, U, []).

eat_until([HIT], H, Acc) -> {lists:reverse(Acc), T};
eat_until([HIT], U, Acc) when H /= U -> eat_until(T, U, [H|Acc]);
eat_until([], _, Acc) -> {lists:reverse(Acc), []}.

url_decode([Hi, Lo | Tail]) ->
    Hex = yaws:hex_to_integer([Hi, Lo]),
    [Hex | url_decode(Tail)];
url_decode([Hi | Tail]) ->
    %% Don't decode the query string here, that is parsed separately.
    [Hi | url_decode(Tail)];
url_decode([HIT]) when integer(H) ->
    [H | url_decode(T)];
url_decode([]) ->
    [];
%% deep lists
url_decode([HIT]) when list(H) ->
    [url_decode(H) | url_decode(T)].

path_norm(Path) ->
    path_norm_reverse(lists:reverse(Path)).

path_norm_reverse("/") ++ T -> start_dir(0, "/", T);
path_norm_reverse(T) -> start_dir(0, "", T).

start_dir(N, Path, "..") -> rest_dir(N, Path, "");
```


Debugging and Launching

- Specifying runtime
 - Multiple runtimes
 - Different for IDE / building / running
- Project properties
- Launch Configurations
- Debugging
 - Starting project

Erlang debugger in Eclipse

The screenshot shows the Eclipse IDE with the Erlang debugger. The main window displays the source code of `zipx.erl` with a breakpoint set at the `get_central_dir` function. The Variables panel shows the current state of variables:

Name	Value
In0	file_descriptor#{module=prim_file, d...
RawIterator	#Fun<erlide_dbg_ieval.13.6345951>
Input	#Fun<erlide_dbg_ieval.14.99645753>
B	<<0,0,0,0,27,0,27,0,56,7,0,0,171,97>
In1	file_descriptor#{module=prim_file, d...
E OCD	read#{disk_name=0, start_disk_name...

The Console panel shows the execution of the `zipx:t` function:

```
Erlang_debug [Erlang application] erlang_debug
Eshell V5.7.2 (abort with ^G)
(erlang_debug@gnu-mac-5.du.uab.ericsson.se)1> zipx:t("/Users/jakob/Desktop/midi.zip").
```


Exercise 2

Navigation 1

- Erlang outline
 - Filtering
 - Navigation
- Erlang navigator
 - Compare w. Package explorer
- Open
 - External function calls
 - Records and macros

Navigation 2

- Quick open (ctrl-O)
- Open module (ctrl-shift-M)
- Searching
- External files

Big projects

- Outline
 - Filtering is a good thing
- Ctrl-click to open
 - Navigation history (as in browser)
- Quick open
- Eclipse text search

Settings

- Setting runtime
- Erlang project properties
- Importing projects

Demo

External Resources

- <http://erlide.org>
Installation instruction, documentation, release notes, mailing list
- <http://github.com/erlide>
Git repo with source, wiki, new features
- <http://www.assembla.com/spaces/erlide>
Issue tracker for the open source project